IN THE TITLE

Please insert the title - - A METHOD AND APPARATUS FOR MANAGING FUNCTIONS - -.

IN THE SPECIFICATION

Please delete the paragraph on page 27, beginning at line 13.

Please replace the paragraph at page 12, line 1, with the following rewritten paragraph:

In addition, Figure 1B shows metadata objects 120A-i associated with underlying functions. Particularly, metadata objects 120A and B respectfully identify action units 105A and B. Additionally, Figure 1B shows metadata object 120C directly stores one of the functions (also referred to as an internal function), and therefore does not need an action unit. Among other things, a metadata object can store information describing the input and output "parameter kinds" to it's underlying function. It is worthwhile to note that the term parameter kind is not used herein as synonymous with data type. Rather, the phrase parameter kind refers to a type of information (e.g., name, street address, Id). Thus, two different parameter kinds (e.g., name and street address) may share the same data type (e.g., string).

Please replace the paragraph beginning at page 12, line 10, with the following re-written paragraph:

A metadata object typically does not contain the values to be used as input parameters to the underlying function. Rather, an execution object is

formed to maintain a context (e.g., input parameter values and resulting output parameter values) for an underlying function. As such, one or more execution objects may be associated with each METADATA object. Figure 1B shows EXECUTION objects 125A-x – 125i-x. Particularly, Figure 1B shows EXECUTION objects 125A-A through 125A-I identifying METADATA object 120A, EXECUTION object 125B identifying METADATA object 120B, EXECUTION object 125C identifying METADATA object 120C, and EXECUTION object 125i identifying METADATA object 120i.

A2 Cncld.

Please replace the paragraph beginning at page 14, line 1, with the following re-written paragraph:

With regard to the set of keys used to distinguish the parameter kinds, each function being tracked by the integration layer can have one or more input parameters and one or more output parameters. A naming convention is used for the different parameter kinds used by the functions. Different functions may share one or more of the same parameter kinds. To illustrate, consider the previous example where function A has as parameters an Id and a street address, while function B has as parameters a name and a street address. In this example, functions A and B share the street address parameter (they share the same parameter kind). In addition, this example has three parameter kinds (Id, street address, and name).

A3

Please replace the paragraph beginning at page 23, line 17, with the following re-written paragraph:

The EXECUTION class 900 of Figure 9 includes the following structures: NAME; CLASS_LABEL; PARAMS; METADATA_OBJECT_PTR; and MANAGER_PTR. As previously described, each execution object is associated with a metadata object. In one embodiment, the NAME structure of an execution object contains the same data as the name structure of the METADATA object to which it is associated. In alternative embodiments, the NAME structure need not store the same data, but rather a name to name look-up structure is used. The CLASS_LABEL structure of the EXECUTION class 900 is used for the same purpose as the CLASS_LABEL structure of the METADATA class.

A4

Please replace the paragraph beginning at page 26, line 18, with the following re-written paragraph:

Figures 11A-B are block diagrams illustrating two additional classes of the integration layer of Figure 1A according to one embodiment of the invention. Figure 11A is a block diagram illustrating a context class 1100 according to one embodiment of the invention. The context class includes the following structures: NAME; KEY_VALUE_COLLECTION; and EXECUTION_COLLECTION.

A5

Please replace the paragraph beginning at page 27, line 22, with the following re-written paragraph:

Figure 11B is a block diagram illustrating a MANAGER class according to one embodiment of the invention. A manager object can be used for managing the execution objects. The MANAGER class 1110 includes the following

A6 cmt

A6 Cncld. structures: CONTEXT_COLLECTION; DEFAULT_CONTEXT; EXECUTION_COLLECTION; and EXECUTION_PATH_COLLECTION.

On page 28 beginning at line 18, please insert the following paragraph:

The EXECUTION_PATH_COLLECTION structure is an ordered collection of level objects used to store a selected execution path. An ordered collection is one that includes data to indicate an order to the object stored therein. A level object is a collection of execution objects for a given level of an execution path. With reference to the previous example shown in Figure 21, the execution path (A→ B→ D→ E→) would be made up of four level objects (one per function) E. A more complex execution path can have more than one execution object per level of execution path, and EXECUTION_PATH_COLLECTION structure is further described later herein.

A7

Please replace the paragraph beginning at page 52, line 10, with the following re-written paragraph:

Figure 20 is a block diagram illustrating the PARAMETER_SATISFY method according to one embodiment of the invention. The PARAMETER_SATISFY method receives the same inputs as the FIND_BY_NEED method previously described (see block 1700). From block 2000, control passes to block 2005.

A8

Please replace the paragraph beginning at page 53, line 17, with the following rewritten paragraph: